A Skip-Connected Evolving Recurrent Neural Network for Data Stream Classification under Label Latency Scenario

Monidipa Das, Mahardhika Pratama, Jie Zhang, Yew Soon Ong
School of Computer Science and Engineering, Nanyang Technological University, Singapore
{monidipadas, mpratama, zhangj, asysong}@ntu.edu.sg

Abstract

Stream classification models for non-stationary environments often assume the immediate availability of data labels. However, in a practical scenario, it is quite natural that the data labels are available only after some temporal lag. This paper explores how a stream classifier model can be made adaptive to such *label latency scenario*. We propose SkipE-RNN, a *self-evolutionary* recurrent neural network with dynamically evolving *skipped*-recurrent-connection for the best utilization of previously observed label information while classifying the current data. When the data label is unavailable, SkipE-RNN uses an auto-learned mapping function to find the best match from the already known data labels and updates the network parameter accordingly. Later, upon availability of true data label, if the previously mapped label is found to be incorrect, SkipE-RNN employs a *regularization technique* along with the parameter updating process, so as to *penalize* the model. In addition, SkipE-RNN has inborn power of *self-adjusting* the network capacity by growing/pruning hidden nodes to cope with the evolving nature of data stream. Rigorous empirical evaluations using synthetic as well as real-world datasets reveal effectiveness of SkipE-RNN in both *finitely* delayed and *infinitely* delayed data label scenarios.

Introduction

Data streams are primarily unbounded and evolving data that continuously arrive over time (Ditzler and Polikar 2013). Because of the two interesting characteristics, namely 'infinite length' and 'evolving nature' (indicating concept drift, concept evolution, etc.), the streaming data classification has received considerable research attention in present years (Masud et al. 2012). Besides, in a typical real-world scenario, it often becomes hard to acquire labeled data due to various issues, including high labelling cost, lack of expertise, confidentiality, etc. Thus, in practice, the streaming data labels become available only after a certain amount of *delay* which may vary from 0 to ∞ (Souza et al. 2015a). However, majority of the existing data stream classification techniques are based on supervised learning models, and hence, their performances are subjected to the availability of the data labels. The same is also true for online active learning models (Ksieniewicz et al. 2019) which require the data labels to be immediately available upon request. Even, the semi-supervised learning principle (Haque, Khan, and Baron 2016) is also not truly effective in the delay scenario of ***online*** learning, since each delayed-labeled sample is always treated as the unlabeled sample as it appears with no label at the beginning, and in infinite delay scenario, the actual labels of the samples are never available to guide the classifier (Souza et al. 2015b). As a consequence, the traditional procedure for monitoring error rate and adapting with concept drift remains no longer applicable. Rather, it becomes necessary to devise a classifier model that should adapt itself by effective utilization of the most recent and/or the most relevant labeled data, even when the true label at the current instant is unavailable.

Contributions: Motivated by the limitations of the state-of-the-art techniques, in this paper we aim to propose a *virtually* supervised learning model with an embedded mapping unit which continuously supervises the learning process by suggesting possible data label, even when there is a label latency during stream classification. We achieve this by proposing a skip-connected *evolving* recurrent neural network (SkipE-RNN) following the teacher forcing policy for feature learning though analysis of *temporality in the data*. Our objective of using recurrent neural network model is to exploit the *temporal dependency* of a data stream as the basis for determining the current data labels, which is often ignored by the existing classifiers. The proposed SkipE-RNN is flexible enough to be applied in the infinitely delayed label scenario as well. The key contributions of this work are as follows.

- This paper proposes SkipE-RNN as a *novel variant* of the recurrent neural network model which is well featured for stream classification under *label latency scenario*.
- SkipE-RNN is defined along with an embedded mechanism of *label mapping* that leads the model to *dynamically evolve skip-connection* for recurrent learning, and thus, helps to continue learning under delay context.
- SkipE-RNN features *flexible architecture* and *self-adaptation power*, according to which it starts learning with a single hidden unit and automatically refines the network structure to cope with *evolving* data characteristic.

- The proposed SkipE-RNN is designed to implicitly learn as per the *teacher forcing policy*, so that it can overcome expensive computation and gradient vanishing/exploding issues in traditional RNN learning.

Effectiveness of the proposed SkipE-RNN is validated using two sets of empirical studies. In the **first case**, we evaluate SkipE-RNN in comparison with five recently developed incremental learning models considering *finite time delay* over four popular benchmark datasets, used in stream classification problems. In the **second case**, SkipE-RNN is assessed with respect to *infinite delay scenario* over six other benchmark datasets and in comparison with the state-of-the-art SCARGC model (Souza et al. 2015b) which is typically proposed for infinite label latency scenario. Experimental results reveal that, with the help of dynamically evolving skipped recurrent connection, the adaptive learning of proposed SkipE-RNN is able to achieve promising accuracy with acceptable computation cost in delayed label condition.

Proposed Model: SkipE-RNN

This section provides the conceptual details of SkipE-RNN, with respect to stream classification scenario as stated below.

Problem Formulation

The proposed prediction model is defined over a data stream classification scenario where the data is continuously generated in the form of data chunks $\mathbb{D}_1, \cdots, \mathbb{D}_C$, such that the number of chunks (C) and the distribution of the data are not known apriori. Each chunk may contain one or more data points/samples $\left[x^{(1)}, \cdots, x^{(T)}\right]$ such that $T \geq 1$ and ${x^{(i)}}^\top \in \mathbb{R}^D$ where D denotes the input feature dimension. Further, since the sample size may randomly depend on accumulating data (Gama, Sebastião, and Rodrigues 2013), the classifier needs to be executed based on the '*prequential test-then-train*' protocol, where each data-chunk is first used to test the generative power of the classifier and then exploited to update the parameters (refer to Figure 1).

More significantly, in a typical stream classification scenario, the actual class/data-label ($y^{(t)}$) of any data point $x^{(t)}$ becomes available after certain amount of time delay $\delta \in [0, \infty)$, which is also termed as *label latency* or verification latency (Souza et al. 2015b). In an extreme case, the labels of the data never arrive after the classification model is initialized. This is called as *infinite label latency* or extreme verification latency. Since in the infinite delay condition there is complete lack of labeled data during classification phase, in order to maintain the definition of a classification problem, it is presumed that small amount of labeled data is available *before* the classification starts. This *initially labeled data* (Dyer, Capo, and Polikar 2014) provides an overall information of the number of classes and the respective feature spaces to support the classification.

According to the problem scenario, when applied in a stream classification environment, the proposed SkipE-RNN is first tested as per the 'prequential-test-then-train' policy. Subsequently, it learns in an *online* and *single-pass* manner during the training phase (see Figure 1).

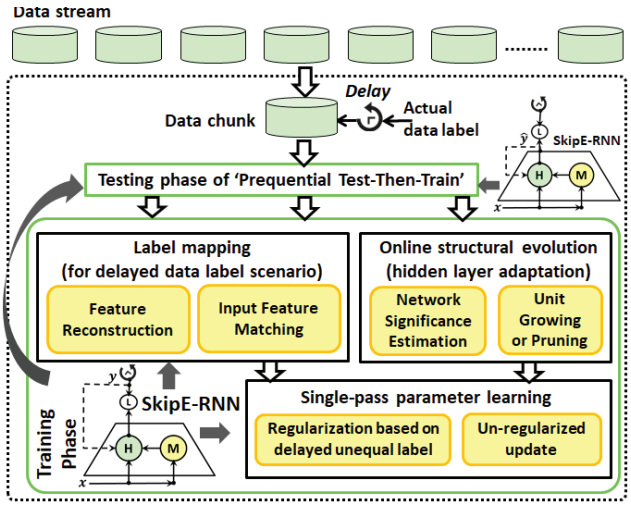

Figure 1: SkipE-RNN in stream classification scenario

SkipE-RNN Architecture

The recurrent architecture and corresponding unfolded computational graph of SkipE-RNN are depicted in Figure 2. As shown in the figure, the recurrent learning module of SkipE-RNN consists of two major units, namely i) *label mapping unit* ('M'), and ii) *feature learning unit* ('H').

Label Mapping Unit The prime objective of this unit is to map the input feature of the current unlabeled sample to the input feature of observed samples with known data labels, and accordingly, determining the most likely label of the current sample. Mathematically, this can be expressed as $M : \mathbb{R}^D \rightarrow \mathbb{I}^+$ and $M({x^{(t)}}^\top) = k^{(t)}$, where $k^{(t)} < (t - \delta)$ indicates the timestamp at which the data label is most likely to be similar as that of the current sample, δ is the temporal delay, $\mathbb{I}^+$ is the set of positive integers, and $\mathbb{R}$ is the set of real numbers. In order to accomplish this mapping, we use an *autoencoder model* (see top-right of Figure 2), which, when applied on $x^{(t)}$, generates a reconstructed input feature $x'^{(t)}$. Later, $x'^{(t)}$ is matched with the previously observed samples with known data labels, and finally the label corresponding to the observed sample $x^{(k^{(t)})}, k^{(t)} < (t - \delta)$ is selected as the best match for the current sample $x^{(t)}$, such that

$$k^{(t)} = \underset{i<t-\delta}{argmin} \quad ||x'^{(t)} - x^{(i)}||_2 \tag{1}$$

The idea is illustrated in Figure 2. During the matching process, instead of comparing with the raw input feature $x^{(t)}$, we use the reconstructed input feature $x'^{(t)}$ to make the mapping unit more generic towards variants of input features associated with the same concepts. This eventually helps the model in handling virtual drift. As shown in Figure 2, the label mapping unit learns in parallel with the feature learning unit for supplying the best $k^{(t)} < (t - \delta)$ value, and thus, results in no additional cost in terms of computational time.

Feature Learning Unit This unit aims at exploiting the temporal dependency (which can be observed though auto-

correlation analysis) of the data stream to recurrently learn the complex features and classify the current sample at time t. Since the actual data labels appear with delay (δ), the idea is to utilize parameters learnt for the known or *initially labeled samples* (Dyer, Capo, and Polikar 2014) with similar input feature, and accordingly ***evolve*** the recurrent connection with δ or more skips, instead of employing usual recurrent connection to the network state at $(t-1)$. The unit takes as input the $k^{(t)} < (t-\delta)$ value as generated by the label mapping unit and establishes the recurrent connection with the network state at the timestamp $k^{(t)}$, while ***skipping*** the intermediate states from $\left(k^{(t)}+1\right)$ to $(t-1)$. Thus, unlike a vanilla RNN, the recurrent connection here is not necessarily defined in terms of immediate precursor in the timeline. The connection involves skipping of network states which is decided ***dynamically*** in accordance with the k value, as received from the label mapping unit (see Figure 2). For

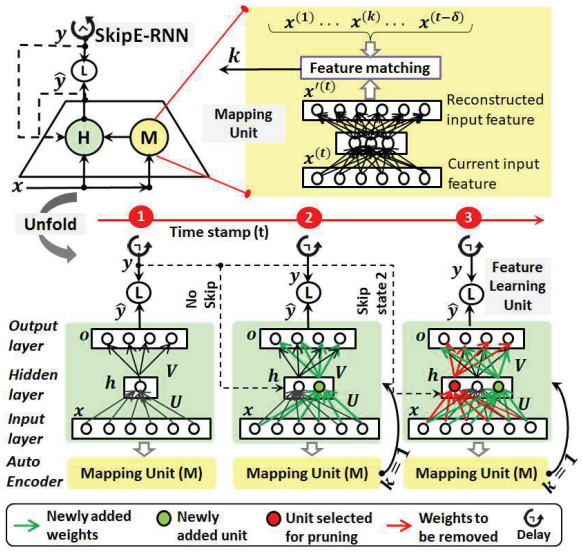

Figure 2: SkipE-RNN architecture

each time step t, the input to feature learning unit is $x^{(t)}$, the hidden layer activation is $h^{(t)}$ (where $h^{(t)\top} \in \mathbb{R}^m$, m is the number of hidden units), un-normalized output is $o^{(t)}$ which is further updated through *softmax* layer to achieve the predicted output $\widehat{y}^{(t)}$ (where $\widehat{y}^{(t)\top} \in \mathbb{R}^S$, S is the number of classes), and the loss is $L^{(t)}$. The input-to-hidden and hidden-to-output connections are parameterized by weight matrices $\mathbf{U}_{[m\times D]}$ and $\mathbf{V}_{[S\times m]}$, respectively. To be noted, the ***recurrent connection*** of the feature learning unit (denoted by the dotted arrows in Figure 2) is not defined by explicit weight matrix. The *output-to-hidden* recurrent connection is maintained implicitly, because of using *hyperplane-based activation* in the hidden layer (refer to the section below).

Parameter Learning

The parameter learning of both label mapping unit and feature learning unit are performed by *single scanning* of data. The parameter learning for the autoencoder model within label mapping unit follows a standard convention, where we use sigmoid activation in the hidden layer and linear activation in the output layer with quadratic loss measured as $\frac{1}{2D}\sum_{i=1}^{D}\left(x_i{}^{(t)} - x_i'^{(t)}\right)^2$. Hence, in this section, we focus on feature learning unit, where the parameter learning is built upon the concept of *skipped recurrent connection* with hyperplane activation (Ferdaus et al. 2019) during forward propagation, followed by *regularized weight updating* during backward propagation.

Forward-propagation: Typically, the hyperplane activation applied on hidden layer is expressed as follows.

$$h^{i(t)} = exp\left(-\eta\frac{d^{i(t)}}{max\left(d^{i(t)}\right)}\right) \tag{2}$$

$$d^{i(t)} = \frac{|y^{(k^{(t)})}|_1 - S\cdot\left(b^i + U_i x^{(t)}\right)}{\sqrt{1+\sum_{j=1}^{D}U_{ij}^2}} \tag{3}$$

where, $d^{i(t)}$ is the distance from $k^{(t)}$-th data point to the i-th feature in the current feature hyperplane at the hidden layer, $|\mathbf{y}|_1$ is the 1-norm of $\mathbf{y}$, S is the output dimension, $b^\top \in \mathbb{R}^m$ is equivalent to the bias associated with the added dimension to the feature plane, and η controls the strength of activation. Subsequently, the predicted class $\widehat{y}^{(t)}$ is estimated as $\widehat{y}^{(t)} = softmax(o^{(t)})$, where $o^{(t)} = c + Vh^{(t)}; c^\top \in \mathbb{R}^S$ is the bias for the output layer. While testing, y and $k^{(t)}$ in Eq. 3 are replaced by $\widehat{y}$ and $(t-1)$, respectively.

The *hyperplane activation* plays two significant roles to establish the dynamically evolving skipped recurrent connection. ***First***, this leads the hidden layer to be *implicitly* influenced by the output $y^{k^{(t)}}$ from an earlier time stamp, thus forming an output-to-hidden recurrent connection, without using explicit weight matrix. This not only reduces the parameter requirement, but also helps the model to directly learn from exact output of earlier time stamps, thus following teacher forcing policy (Goodfellow et al. 2016). Because of the *teacher forcing*, the model can also avoid the overhead of *back-propagation through time* and the exploding/vanishing gradient issues. ***Second***, this helps the model recurrently learn from the network state at an earlier time stamp $k^{(t)} < (t-\delta)$ which is dynamically decided by the label mapping unit, and eventually results in a skip-connected recurrent architecture. Thus, the model can continuously learn from the streaming data without waiting for the true data label, which fits well with the label latency scenario.

Regularized Weight Updating: The weight updating process of SkipE-RNN feature learning unit is governed by the following cost function:

$$L\left(y_i^{(k^{(t)})}, \widehat{y}_i\right) = -\sum_i y_i^{\left(k^{(t)}\right)}\cdot log(\widehat{y}_i) + \mathcal{R} \tag{4}$$

where $\widehat{y}$ is the predicted label, $y^{(k^{(t)})}$ is the known/observed label at $k^{(t)}$, and $\mathcal{R}$ is a ***regularization term***. The definition of $\mathcal{R}$ at each timestamp t changes dynamically as follows.

$$\mathcal{R} = \frac{\lambda}{2}\left[||V^{(t)} - V^{(t-\delta)}||_2 + ||U^{(t)} - U^{(t-\delta)}||_2\right], \quad \text{if} \quad y^{(t-\delta)} \neq y^{(k^{(t-\delta)})}$$
$$= 0, \quad otherwise \tag{5}$$

Here, λ denotes the regularization parameter. The key idea here is, if the actual label $y^{(t-\delta)}$ of a sample at time $(t-\delta)$ is revealed at current time t (i.e. after the end of the delay period δ) and is found to be mismatched with the suggested/mapped label $y^{(k^{(t-\delta)})}$, then ***penalize the model*** by bringing the parameter states back to what these were earlier and then proceed for the current updating process. Otherwise, the regularization term (penalty) is kept as zero.

In case of infinite delay ($\delta = \infty$), the actual label is never available for validating the correctness. So, in such case, the model continues learning only based on the supervision from label mapping unit, without any penalty (i.e. $\mathcal{R}$=0).

The recursive computation of gradient starts with $\frac{\partial L}{\partial L^{(t)}} = 1$. Subsequently, the gradients of weight parameters are estimated as per regularization condition, in following manner.

$$\nabla_V L = (\nabla_{o^{(t)}} L) \cdot \left(h^{(t)}\right)^\top + \psi\lambda\left(V - V^{(t-\delta)}\right) \tag{6}$$

$$\nabla_U L = \left(\left(\frac{\eta}{A} \cdot h^{(t)}\right) \circ \left(\nabla_{h^{(t)}} L\right)\right) \cdot x^{(t)\top} + \psi\lambda\left(U - U^{(t-\delta)}\right) \tag{7}$$

where, $\left(\nabla_{o^{(t)}} L\right)_i = \widehat{y}_i^{(t)} - y_i^{(k^{(t)})}$, $\nabla_{h^{(t)}} L = V^\top \left(\nabla_{o^{(t)}} L\right)$, and A is the maximum of the data-sample distances to the hidden layer features. In the Eqs. 7 and 6, $\psi = 1$ when $y^{(t-\delta)} \neq y^{(k^{(t-\delta)})}$, otherwise $\psi = 0$.

Hidden Layer Adaptation

The structural adaptation within the hidden layer of SkipE-RNN is achieved based on network significance (NS) (Ashfahani and Pratama 2019) which represents the generalization power of the network in terms of bias and variance as follows: $NS = Bias^2 + Var$. In case of the SkipE-RNN, $NS = (E[o] - y)^2 + (E[o^2] - E[o]^2)$ where, $E[o]$ is the expectation of un-normalized output from feature learning unit. As per SkipE-RNN parameter learning, $E[o] = \int_{-\infty}^{\infty}(c + V \cdot h)p(h)dh = c + V \cdot E[h]$, where $E[h] = e^{\left(-\frac{E(d)}{max(E(d))}\right)}$ and $E[d^i] = (|y^{(k)}|_1 - \left(b^i + U_i \cdot \mu\right))/(\sqrt{1 + \sum_{j=1}^{D} U_{ij}})$; μ is the mean of data distribution.

A hidden node is added to the hidden layer when we detect high bias, indicating model-underfit, and thus requirement for enlarging structural complexity. The high bias condition is represented as: $\mu_{Bias}^t + \sigma_{Bias}^t \geq \mu_{Bias}^{min} + \pi\sigma_{Bias}^{min}$. Here, μ_{Bias}^t, σ_{Bias}^t are respectively the mean and the standard deviation of bias at the t-th time instance and μ_{Bias}^{min}, σ_{Bias}^{min} are that for the minimum bias till the time instance t. Once a new hidden node is added to the layer, the associated parameters (b, U, V) can be set by employing adaptive scope selection mechanism (Wang and Li 2017; Tang et al. 2017). Similarly, a hidden node is pruned from the hidden layer when we detect high variance, indicating model-overfit, and thus, requirement for reducing structural complexity. The high variance condition is represented as: $\mu_{Var}^t + \sigma_{Var}^t \geq \mu_{Var}^{min} + 2\chi\sigma_{Var}^{min}$, where μ_{Var}^t, σ_{Var}^t are respectively the mean and standard deviation of the variance at the t-th time instance, and μ_{Var}^{min}, σ_{Var}^{min} are that for the minimum variance till the time instance t. Once high variance is detected, the p-th hidden unit, associated with the least ever average activation value in the hidden layer is chosen for pruning. Thus, $p = \underset{i}{argmin}\left(\lim_{T\to\infty}\sum_{t=1}^{T}\frac{h^{i(t)}}{T}\right)$. The π and χ are dynamic constants controlling the confidence level of the sigma rule (see the high bias/variance condition equations above) and are estimated as $1.3\exp(-Var) + 0.7$. A typical scenario of hidden unit growing and pruning within feature learning unit of SkipE-RNN is depicted in Figure 2.

Experimental Evaluation

We evaluate the model using two case studies, covering *finitely delayed* and *infinitely delayed* label situation, respectively. SkipE-RNN and other baselines are executed under the same 'prequential test-then-train' environment. The source code of SkipE-RNN is available online [1].

Table 1: Specifications for the datasets used in Case Study-1

Datasets	Specifications				
	#Instance	#Attr.	#Target	#Task	Characteristics
ELECT.	45312	8	2	45	Real, non-stationary with covariate drifts
HYPER.	120000	4	2	120	Synthetic, non-stationary with gradual concept drifts
SEA	100000	3	2	100	Synthetic, non-stationary with recurring drifts
P-MNIST	70000	784	10	70	Synthetic, non-stationary with recurring drifts

Case Study-1: Finite Latency Scenario

Background The case study-1 is prepared for the finite label latency scenario considering the delay $\delta \in [0, 500]$. However, as discussed in our introductory section, the existing supervised, semi-supervised, and active learning models are not fit for ***online*** delay scenario. Hence, first we set $\delta = 0$ and compare our proposed model with the state-of-the-art stream classification techniques to evaluate its power of handling evolving data streams under usual online learning environment. Subsequently, we consider different δ value to assess the effect of delay on our proposed model performance.

Datasets In case study-1, we evaluate SkipE-RNN using four popular data streams: *i) Electricity-pricing* (Ditzler and Polikar 2013), *ii) Hyperplane* (Bifet et al. 2010), *iii) Sea* (Street and Kim 2001), and *iv) Permuted-MNIST* (Lopez-Paz and others 2017). A summary of these is presented in Table 1. Electricity-pricing, Hyperplane, and Sea are well-used as data streams with variants of concept drifts (refer to Table 1). Further, these show prominent temporal dependency which are appropriate for evaluating the effectiveness of our proposed recurrent model. We also use Permuted (P)-MNIST to evaluate the models against high input dimension. This is a variant of MNIST dataset which is rebuilt through digit-transformation by permuting the pixels. The P-MNIST

[1] https://github.com/SkipE-RNN/Share

dataset is widely used to assess continual learning ability of the models in data stream classification scenario.

Baselines Data stream classification needs to take care of several issues, including *one-pass learning, limited computational resources, temporal dependency*, and *self adaptation to evolving data*. We, therefore, have chosen the following baselines dealing with one or more of these aspects.

- **PNN** (Rusu et al. 2016): Progressive neural network; Primarily deals with concept evolution in a data stream;
- **DEN** (Lee et al. 2017): Dynamically expandable network; Extension of PNN; Puts forward selective retraining, dynamic expansion, and splitting/duplicating methods;
- **HAT** (Serrà et al. 2018): Task-based hard attention mechanism; Developed for concept evolution in a data stream;
- **Learn++.NSE** (Ditzler and Polikar 2013): Appropriate for dealing with variants of drift scenarios; Capable of learning in non-stationary environments;
- **Online Multiclass (OMC) Boosting** (Jung, Goetz, and Tewari 2017): Variant of online boosting algorithm; Uses optimal number of classifier in the ensemble to achieve desired accuracy with reduced computational cost;
- **RNN_tanh**: Vanilla RNN model (Goodfellow et al. 2016) with single layer using $tanh$ activation and learning based on back propagation through time (BPTT); This comparison is necessary, since SkipE-RNN is a variant of RNN.

Hyper-parameter settings For each baseline, we considered the same configuration as given in the respective papers/source-codes. If the performance is surprisingly poor, we fine-tuned the same to record the best ones. In case of SkipE-RNN, the activation control constant $\eta = 0.5$ and $\lambda = 0.001$. However, since SkipE-RNN is self-evolving, there is no hyper-parameter regarding its structural setup.

Results and Discussions The model performance is measured using four criteria: *classification rate* (**CR**), *parameter count* (**PC**), *hidden unit count* (**HU**), and *execution time* (**ET**). The results of comparative study are presented through Table 2. The values are recorded as average of *five* random seeds. For OMC boosting and Learn++.NSE, PC=NA and HU=NA, since these are based on decision tree model. On analyzing the results, we can infer the following.

Comparison on classification rate (CR): Even with the minimal usage of parameters, the proposed SkipE-RNN is able to achieve comparable and sometimes substantially better classification accuracy for each considered dataset.

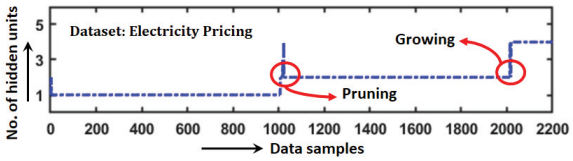

Figure 3: Typical hidden unit adjustment in SkipE-RNN

Table 2: Performance study under finite latency scenario

Data	Models	Metrics				WT
		CR	HU	PC	ET(s)	
Electricity	PNN	57.84 ± 4.52	78	1868	51.345	X
	DEN	56.54 ± 7.66	16	178	72.54	X
	HAT	56.63 ± 8.04	20	242	145.57	X
	OMC-Boosting	77.12 ± 4.57	NA	NA	56.80	X
	Learn++.NSE	77.18 ± 9.3	NA	NA	169.88	X
	RNN_tanh	65.12± 7.21	4	104	35.41	X
	SkipE-RNN$_{\delta=0}$	**77.75 ± 09.87**			19.22	
	SkipE-RNN$_{\delta=50}$	75.01 ± 11.13	4	46	30.92	
	SkipE-RNN$_{\delta=500}$	74.90 ± 10.29			24.95	
Hyperplane	PNN	85.07 ± 7.12	42	560	190.196	X
	DEN	91.83 ± 4.17	8	58	202.57	X
	HAT	77.9 ± 10.76	12	98	370.8	X
	OMC-Boosting	86.18 ± 3.73	NA	NA	111.74	X
	Learn++.NSE	90.35 ± 2.48	NA	NA	374	X
	RNN_tanh	76.55 ± 2.82	4	80	101.58	X
	SkipE-RNN$_{\delta=0}$	**92.45 ± 2.76**			39.66	
	SkipE-RNN$_{\delta=50}$	90.95 ± 3.55	2	16	73.78	
	SkipE-RNN$_{\delta=500}$	91.86 ± 2.43			68.47	
Sea	PNN	84.87 ± 6.52	33	353	152.46	X
	DEN	79.95 ± 19.28	6	38	169.72	X
	HAT	74.65 ± 10.1	10	72	327	X
	OMC-Boosting	87.86 ± 3.85	NA	NA	77.44	X
	Learn++.NSE	90.17 ± 5.96	NA	NA	268	X
	RNN_tanh	75.17 ± 2.94	4	42	105.22	X
	SkipE-RNN$_{\delta=0}$	**91.17 ± 6.69**			33.81	
	SkipE-RNN$_{\delta=50}$	88.65 ± 6.87	2	14	63.06	
	SkipE-RNN$_{\delta=500}$	87.53 ± 7.94			57.34	
P-MNIST	PNN	64.42 ± 8.77	260	170K	152.95	X
	DEN	52.08 ± 22.6	440	290K	399.83	X
	HAT	59.64 ± 18.88	60	24.9K	207.04	X
	OMC-Boosting	35.58 ± 20.51	NA	NA	5K	X
	Learn++.NSE	NA	NA	NA	NA	NA
	RNN_tanh	69.62 ± 11.36	250	198K	925.10	X
	SkipE-RNN$_{\delta=0}$	**84.01 ± 13.61**			549.76	
	SkipE-RNN$_{\delta=50}$	81.49 ± 13.40	79	63K	592.62	
	SkipE-RNN$_{\delta=500}$	81.66 ± 13.56			596.24	

X: Reject null hypothesis that a model performs better than SkipE-RNN

The average percentage improvement of SkipE-RNN over fixed structured vanilla RNN is more than 13%, and also, it is achieved with notably less execution time. The modelling of temporal dependencies through the dynamically adaptive recurrent architecture helps SkipE-RNN to attain this encouraging performance. The significantly higher CR of SkipE-RNN for P-MNIST dataset further demonstrates its potential to deal with high dimensional, unstructured data.

Comparison on parameter count (PC) and hidden unit (HU) requirement: Though the PC and HU for SkipE-RNN are sometimes higher (e.g. in case of P-MNIST), these are not prefixed from the beginning. As depicted in Figure 3, the execution of SkipE-RNN starts with only one unit in a single hidden layer, and then, gradually the number of hidden units is adjusted (added or removed) from the layer so as to cope up with the time varying distribution and conceptual drift of the streaming data. This gradual and on-the-fly structural adjustment helps SkipE-RNN to achieve desired accuracy with optimal number of parameters.

Comparison on execution time (ET): Table 2 also shows that even with the dynamic layer adaptation overhead, SkipE-RNN is able to achieve acceptable accuracy within reasonable time. Though online multiclass (OMC) boosting is in general popular for low computational time requirement, our proposed self-adaptive SkipE-RNN is found to be even faster than OMC-Boosting for each considered dataset. This is so because SkipE-RNN is based on single classifier

and it works based on optimal no. of parameters.

Sensitivity on label latency: Effect of different (finite and non-zero) latency duration on SkipE-RNN performance is also summarized in Table 2. The result shows that the classification performance of SkipE-RNN is not very sensitive to the latency/delay length (δ). The average degradation in prediction accuracy is approximately 2.5% with respect to no-delay situation. This demonstrates the effectiveness of using the label mapping unit along with the recurrent feature learning, and also proves the usefulness of regularizing the model in case of label mismatch in subsequent phases.

Results of statistical tests: The performance of SkipE-RNN is also validated using ***Wilcoxon statistical test***, as summarized in Table 2 (see **WT** column). The results validate that, in every case, the proposed SkipE-RNN performs statistically better or similar compared to the other considered models (with a significance level of 5%).

Case Study-2: Infinite Latency Scenario

Background In this case, the actual label of the data is never available after the classification starts, and thus, $\delta = \infty$ here. However, as per the state-of-the-art convention (Souza et al. 2015b; Dyer, Capo, and Polikar 2014), the scenario assumes that a small amount of initial labeled data with at least one sample for each possible class is available before the classification begins. This is necessary to maintain the definition of a classification problem.

Datasets Case study-2 is carried out using *five* synthetic benchmark datasets, typically prepared for infinite delay scenario, and one more real-world problem (refer Table 3). All these datasets are collected from the website (Souza et al. 2019) of our main competitor method SCARGC (Souza et al. 2015b). The synthetic datasets i.e. 1CHT, 2CHT, GEARS-2C-2D, MG-2C-2D, and UG-2C-5D show incremental concept drift, whereas the real-world dataset NOAA shows real and recurrent drift. For all the datasets, we have considered the amount of ***initial labeled*** data to be $<2\%$ with ***equal distribution*** of each class.

Table 3: Dataset description for the Case Study-2

Dataset	#Class	#Feature	Drift	Length	Type
1CHT	2	2	400	16000	Synthetic
2CHT	2	2	400	16000	Synthetic
GEARS_2C_2D	2	2	2000	200000	Synthetic
MG_2C_2D	2	2	2000	200000	Synthetic
UG_2C_5D	2	5	2000	200000	Synthetic
NOAA	2	8	180	18000	Real

Baselines We compare our model against the state-of-the-art SCARGC (*S*tream *C*lassification *A*lgorithm *G*uided by *C*lustering) (Souza et al. 2015b), which is primarily proposed for handling infinitely delayed label context. SCARGC is extremely efficient in terms of running time. However, this significantly depends on clustering phase and requires user knowledge for parameter setting. In addition to SCARGC, as suggested by Souza et al., we also consider two *bounds*, namely *Static* and *Sliding*, as the baselines for comparative study. The *Static* classifier is only trained on the given initial samples (with known label) and is not updated over time. So, its performance is equivalent to the lower bound, indicating how well the initial samples represent the whole dataset and whether classifier updating is necessary. Contrarily, after training on the initial samples, the *Sliding* model is constantly updated whenever a new sample comes while dropping the old ones. Thus, the performance of this model is expected to be nearly an upper bound.

To be noted, the *arbitrary sub-population tracker* (APT) (Krempl 2011) and *compacted object sample extraction* (COMPOSE) (Dyer, Capo, and Polikar 2014) are two other state-of-the-art classifiers proposed for infinitely delayed label scenario. However, it is clearly demonstrated by Souza et al. 2015b that SCARGC is able to offer similar or even better accuracy with significantly reduced computational time, compared to APT and COMPOSE, for these datasets. We, therefore, have compared SkipE-RNN with SCARGC only.

Hyper-parameter settings To compare with SCARGC, we have considered its 1NN version with the same parameter setup as used by Souza et al. 2015b. SkipE-RNN does not require ad-hoc parameter settings, since it is self-evolving.

Table 4: Accuracy (%) comparison in infinite delay scenario

Dataset	Static	Sliding	SCARGC	SkipE-RNN
1CHT	91.96	99.24	99.31	**99.34**
2CHT	54.03	85.44	86.13	**88.74**
GEARS_2C_2D	93.62	99.86	95.60	**95.61**
MG_2C_2D	49.00	90.40	61.91	**87.31**
UG_2C_5D	68.81	89.91	79.98	**91.92**
NOAA	66.19	72.01	68.37	**73.04**

Table 5: Run-time (s) comparison in infinite delay scenario

Dataset→	1CHT	2CHT	GEARS	MG_2C_2D	UG_2C_5D	NOAA
SCARGC	1.46	1.83	16.10	16.20	19.31	2.27
SkipE-RNN	5.33	5.80	72.27	68.02	71.40	6.84

Results and Discussions The comparative study is carried out with respect to *average accuracy* for classifying each data stream. We also compare the *computational time* of our proposed model against SCARGC. The results of empirical analysis are presented through Tables 4-5 and Figure 4.

Analysis on model accuracy: It can be observed from the Table 4 that for both synthetic and real-world datasets, especially in case of 2CHT, MG-2C-2D, UG-2C-5D, and NOAA, the classification accuracy for SkipE-RNN is substantially high compared to state-of-the-art SCARGC model. Interestingly, the poor performance of *static model* for these particular datasets indicates that the model must include appropriate adaptation quality for classifying these data. Hence, we can conclude that the adaptation power of the proposed SkipE-RNN is far better than that of SCARGC. For a better understanding, we have plotted the classification performance of both SCARGC and SkipE-RNN on 'per step basis' in Figure 4 for some datasets. The

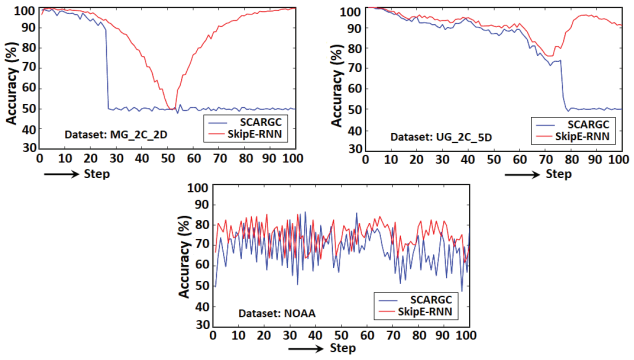

Figure 4: Comparative study of accuracy in the infinitely delayed label scenario

figure shows that though SkipE-RNN sometimes starts with low accuracy at the very beginning, it has powerful enough adaptation mechanism to finally reach the desired level. Judicious learning from initial samples through dynamically evolving skipped-recurrent-connection helps the model to achieve such classification performance.

Analysis on computation time requirement: Apparently, it seems from Table 5 that the time requirement for SkipE-RNN is quite higher compared to SCARGC algorithm. However, when analyzed from the perspective of per sample time requirement, we can find that the proposed SkipE-RNN requires approximately 0.0004 second to process/classify each sample (for any dataset), which is not at all a poor performance indicator in the scenario of online stream classification under infinite label latency. Further, it can be noted from literature that the computational time requirement for SkipE-RNN is substantially less compared to the other state-of-the-art models, like APT (Krempl 2011) and COMPOSE (Dyer, Capo, and Polikar 2014), recently proposed for infinite label latency scenario.

In summary, even if the true labels of the samples are never available during classification, SkipE-RNN still can work under the virtual supervision of its label mapping unit, which continuously suggests for possible labels of the unlabeled samples. Though SkipE-RNN cannot outperform SCARGC in terms of time requirement, it is able to achieve far better classification accuracy within reasonable computation time and without using ad-hoc parameter settings.

Related Works

Evolving Data Stream Classifiers: The existing approaches for evolving stream classification are focused more towards incremental/continual learning (Polikar et al. 2001) than dealing with the issue of label latency. These models are constructed either on a single classifier or on an ensemble of classifiers. Compared to a single classifier model, an ensemble method (Jung, Goetz, and Tewari 2017; Kolter and Maloof 2005; Polikar et al. 2001) can better control the bias-variance problem by providing sufficient diversity of its base classifiers. However, the ensemble-models suffer from notably high structural complexity, which is resolved by single-classifier model (Rusu et al. 2016; Lee et al. 2017; Serrà et al. 2018). Nevertheless, being defined on supervised and active learning principles, most of these models are not suitable for delayed data label scenario. Though the semi-supervised techniques are typically proposed to work based on little amount of labeled data (Wu, Li, and Hu 2012; Haque, Khan, and Baron 2016), these are still not fit for delay context in ***online*** learning. This is because in the online learning environment the streaming samples are required to be immediately classified, but due to the delay, all the samples are appeared to be unlabeled for these algorithms. Moreover, in case of the infinite delay, the actual labels of the samples are never available to guide the classifier.

In order to deal with infinite delay, recently Kremple 2011, Dyer et al. 2014 and Souza et al. 2015b; 2015a proposed some techniques, mostly guided by the concepts of sup-population/cluster or micro-clusters and computational geometries. However, these models either use a number of predefined assumptions (e.g. constant sub-population size) or require ad-hoc parameter settings (e.g. no. of clusters, size of pool etc.) which restrict them in several applications. High computational cost also becomes a major issue for some of these algorithms (Dyer, Capo, and Polikar 2014).

Skip-connected RNNs: RNN models with skip connection have recently gained considerable attention in deep learning research. However, the existing skip-connected RNNs (Campos et al. 2017; Chen et al. 2017; Chang et al. 2017) are mostly designed based on fixed architecture of the feed-forward component, making these unfit for handling the evolving data streams. Though the skip-connected RNN model of Miikkulainen et al. (2019) can evolve its network topology, this model works on epoch-basis and requires immediate access to data labels. Thus, it becomes neither fit for online learning nor fit for delayed label scenario.

Conclusions

Delay in arrival of data label is a common concern in the realm of stream classification. In this paper, we have proposed SkipE-RNN, a variant of RNN, to tackle such situation through its dynamically evolving and skipped recurrent connection which helps the model to continue learning from already known or initially observed samples. The proposed SkipE-RNN-based classification model is applicable for finite as well as infinite label latency scenario, and also, it can handle the evolving characteristics of data stream through the auto-adaptation of its network configuration. Eventually, this becomes appropriate for a wide range of real-life applications. Unlike existing classifiers for delayed label context, our proposed model is exempted from predefined assumptions and ad-hoc parameter settings, while producing promising accuracy within acceptable computational time. In future, we plan to extend SkipE-RNN with added convolution module to deal with complex image streams.

Acknowledgments

This work is financially supported by the NTU-SUTD-ASTAR AI partnership grant (#RGANS1902).

References

Ashfahani, A., and Pratama, M. 2019. Autonomous deep learning: Continual learning approach for dynamic environments. In *SIAM International Conference on Data Mining.*

Bifet, A.; Holmes, G.; Kirkby, R.; and Pfahringer, B. 2010. Moa: Massive online analysis. *Journal of Machine Learning Research* 11(May):1601–1604.

Campos, V.; Jou, B.; Giró-i Nieto, X.; Torres, J.; and Chang, S.-F. 2017. Skip rnn: Learning to skip state updates in recurrent neural networks. *arXiv preprint arXiv:1708.06834*.

Chang, S.; Zhang, Y.; Han, W.; Yu, M.; Guo, X.; Tan, W.; Cui, X.; Witbrock, M.; Hasegawa-Johnson, M. A.; and Huang, T. S. 2017. Dilated recurrent neural networks. In *Advances in Neural Information Processing Systems*, 77–87.

Chen, Y.; Li, J.; Xiao, H.; Jin, X.; Yan, S.; and Feng, J. 2017. Dual path networks. In *Advances in Neural Information Processing Systems*, 4467–4475.

Ditzler, G., and Polikar, R. 2013. Incremental learning of concept drift from streaming imbalanced data. *IEEE Transactions on Knowledge and Data Enggineering* 25(10):2283–2301.

Dyer, K. B.; Capo, R.; and Polikar, R. 2014. Compose: A semisupervised learning framework for initially labeled nonstationary streaming data. *IEEE transactions on neural networks and learning systems* 25(1):12–26.

Ferdaus, M. M.; Pratama, M.; Anavatti, S.; and Garratt, M. A. 2019. Palm: An incremental construction of hyperplanes for data stream regression. *IEEE Transactions on Fuzzy Systems (in press)*.

Gama, J.; Sebastião, R.; and Rodrigues, P. P. 2013. On evaluating stream learning algorithms. *Machine learning* 90(3):317–346.

Goodfellow, I.; Bengio, Y.; Courville, A.; and Bengio, Y. 2016. *Deep learning*, volume 1. MIT press Cambridge.

Haque, A.; Khan, L.; and Baron, M. 2016. SAND: Semi-supervised adaptive novel class detection and classification over data stream. In *Thirtieth AAAI Conference on Artificial Intelligence*, 1652–1658. AAAI Press.

Jung, Y. H.; Goetz, J.; and Tewari, A. 2017. Online multiclass boosting. In *Advances in neural information processing systems (NIPS)*, 919–928.

Kolter, J. Z., and Maloof, M. A. 2005. Using additive expert ensembles to cope with concept drift. In *Proceedings of the 22nd International Conference on Machine Learning*, 449–456. ACM.

Krempl, G. 2011. The algorithm apt to classify in concurrence of latency and drift. In *International Symposium on Intelligent Data Analysis*, 222–233. Springer.

Ksieniewicz, P.; Woźniak, M.; Cyganek, B.; Kasprzak, A.; and Walkowiak, K. 2019. Data stream classification using active learned neural networks. *Neurocomputing* 353:74–82.

Lee, J.; Yun, J.; Hwang, S.; and Yang, E. 2017. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*.

Lopez-Paz, D., et al. 2017. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems (NIPS)*, 6467–6476.

Masud, M. M.; Woolam, C.; Gao, J.; Khan, L.; Han, J.; Hamlen, K. W.; and Oza, N. C. 2012. Facing the reality of data stream classification: coping with scarcity of labeled data. *Knowledge and information systems* 33(1):213–244.

Miikkulainen, R.; Liang, J.; Meyerson, E.; Rawal, A.; Fink, D.; Francon, O.; Raju, B.; Shahrzad, H.; Navruzyan, A.; Duffy, N.; et al. 2019. Evolving deep neural networks. In *Artificial Intelligence in the Age of Neural Networks and Brain Computing*. Elsevier. 293–312.

Polikar, R.; Upda, L.; Upda, S. S.; and Honavar, V. 2001. Learn++: An incremental learning algorithm for supervised neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, part C* 31(4):497–508.

Rusu, A. A.; Rabinowitz, N. C.; Desjardins, G.; Soyer, H.; Kirkpatrick, J.; Kavukcuoglu, K.; Pascanu, R.; and Hadsell, R. 2016. Progressive neural networks. *arXiv preprint arXiv:1606.04671*.

Serrà, J.; Surís, D.; Miron, M.; and Karatzoglou, A. 2018. Overcoming catastrophic forgetting with hard attention to the task. In *35th International Conference on Machine Learning*, volume 80, 4548–4557.

Souza, V. M.; Silva, D. F.; Batista, G. E.; and Gama, J. 2015a. Classification of evolving data streams with infinitely delayed labels. In *2015 IEEE 14th International Conference on Machine Learning and Applications*, 214–219. IEEE.

Souza, V. M.; Silva, D. F.; Gama, J.; and Batista, G. E. 2015b. Data stream classification guided by clustering on nonstationary environments and extreme verification latency. In *Proceedings of the 2015 SIAM International Conference on Data Mining*, 873–881. SIAM.

Souza, V. M.; Silva, D. F.; Batista, G. E.; and Gama, J. 2019. Nonstationary Environment— Archive. https://sites.google.com/site/nonstationaryarchive/. [Online; access Sep-2019].

Street, W. N., and Kim, Y. 2001. A streaming ensemble algorithm (sea) for large-scale classification. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, 377–382. ACM.

Tang, J.; Qiao, J.; Wu, Z.; Zhang, J.; and Yan, A. 2017. Selective ensemble random neural networks based on adaptive selection scope of input weights and biases for building soft measuring model. In *International Conference on Neural Information Processing*, 576–585. Springer.

Wang, D., and Li, M. 2017. Stochastic configuration networks: Fundamentals and algorithms. *IEEE transactions on cybernetics* 47(10):3466–3479.

Wu, X.; Li, P.; and Hu, X. 2012. Learning from concept drifting data streams with unlabeled data. *Neurocomputing* 92:145–155.